

# An efficient approach for Interactive Sequential Pattern Recognition

Jorge Calvo-Zaragoza<sup>a,\*</sup>, Jose Oncina<sup>a</sup>

<sup>a</sup>*Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante, Carretera San Vicente del Raspeig s/n, 03690 Alicante, Spain*

---

## Abstract

Interactive Pattern Recognition (IPR) is an emergent framework in which the user is involved actively in the recognition process by giving feedback to the system when an error is detected. Although this framework is expected to reduce the number of errors to correct, it may increase the time required to complete the task since the machine needs to recompute its proposal after each interaction. Therefore, a fast computation is required to make the interactive system profitable and user-friendly. This work presents an efficient approach to deal with IPR tasks when data has a sequential nature. Our approach includes some computation at the very beginning of the task but it then achieves a linear complexity after user corrections. [We also show how these tasks can be effectively carried out if the solution space is defined with a Regular Language. This fact has indeed proven to be the most relevant factor to improve the efficiency of the approach.](#) Several experiments are carried out in which our proposal is faced against a classical search. Results show a reduction in time in all experiments considered, solving efficiently some complex IPR tasks thanks to our proposals.

*Keywords:* Interactive Pattern Recognition, Sequential Pattern Recognition, Human-Computer Interaction, Word Graph, Efficient search

---

---

\*Corresponding author: Tel.: +349-65-903772; Fax: +349-65-909326  
*Email addresses:* [jcalvo@dlsi.ua.es](mailto:jcalvo@dlsi.ua.es) (Jorge Calvo-Zaragoza), [oncina@ua.es](mailto:oncina@ua.es) (Jose Oncina)

## 1. Introduction

Current Pattern Recognition systems are far from being error-free [1, 2, 3, 4]. If a high or full accuracy is an important issue, an expert supervisor is required to correct the mistakes. Traditionally, these corrections are performed offline: the machine proposes its solution and the supervisor corrects the output off the system error by error. The Interactive Pattern Recognition (IPR) framework involves actively the user in the recognition process so as to reduce the effort needed in the previous scenario [5]. A common IPR task is developed as follows (see Fig. 1):

1. An input is given to the system.
2. The system proposes a solution.
3. If some error is found, the user gives feedback to the system.
4. Taking into account the new information, the system proposes a new solution and returns to the previous step.

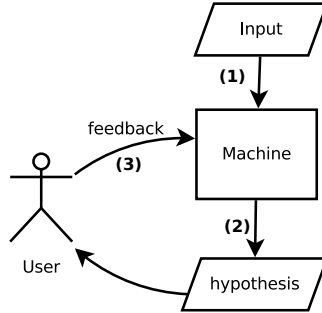


Figure 1: General scheme of an IPR task.

Note that the main goal of IPR is not to make the user learn how to perform such task, which would be more related to fields like Interactive Learning [6], but to complete the work saving as much as possible the available resources.

Including a supervisor in the recognition process provides new ways to improve the efficacy of the system [7]. For instance, corrections provide error-free parts of the solution, which can be helpful to be more accurate in the remaining one, as well as new context-related labeled data.

However, the most important difference when dealing with an IPR task is the performance evaluation. Since the user is considered the most valuable resource, the performance of an IPR system must be related to the user effort  
 25 needed to complete the task. This is commonly measured as the number of user interactions, regardless the nature of them [8].

Theoretically, this framework reduces the number of corrections that would have to be done in a non-interactive scenario. Nevertheless, empirical studies with real users, such as those carried out under Transcriptorium [9] or Cas-  
 30 MaCat [10] projects, showed that the interactive approach may entail some drawbacks from users' point of view. For instance, if the human-computer interaction is not friendly enough or the user is not used to working in an interactive way, the time and/or effort needed to complete the task could even be worse than in the conventional, non-interactive post-editing scenario.

This work focuses on the case of pattern recognition tasks with a sequential  
 35 nature –that is, outputs are sequences of symbols from a discrete alphabet–, which can also be called Sequential Pattern Recognition, or Interactive Sequential Pattern Recognition (ISPR) if they are developed under the interactive framework. Within this context it is often assumed that the user corrects the  
 40 first error found following a specific order (*eg.* left-to-right order). Note that this is not a strong constraint because human interpretation of sequences usually follows this kind of order. We assume that the user acts like an oracle, that is, she is able to detect any error and does know the actual sequence being pursued. It is important to stress that this is not always the real scenario. Reader may  
 45 check the book of Toselli et al. [5] for a broader discussion on this topic.

Table 1 shows an example of this scenario for the Optical Character Recognition (OCR) task over the ESPOSALLES database [11] (the symbol # means user accepts the proposed hypothesis). Note that the first hypothesis contains 4 errors but the interactive approach only needs 2 corrections.

50 Within this context, the main advantage of the left-to-right assumption is that after each feedback, the machine is given an error-free prefix because the correction on a specific symbol implicitly validates the previous ones. Oncina

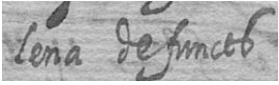
Input	
Output	lena defuncts
Machine	lena defmds
User	u
Machine	lena defunds
User	c
Machine	lena defuncts
User	#

Table 1: Example of first error correction as human feedback in an OCR task (input from the ESPOSALLES database [11]).

[12] developed the optimum strategy to minimize the number of user corrections in this framework, and some works have been already carried out applying this  
55 strategy [13].

This algorithm needs to consider every single hypothesis that is feasible for the task. Current technologies are typically based on structures that compress the space of solutions to make them more manageable, but in which many hypothesis are pruned. Under the interactive case, this provokes that the opti-  
60 mum criterion can not properly followed. In our work we focus on tasks that can be modeled with structures that are able to contain the complete space of hypothesis. Consequently, the decoding complexity increases.

From the practical point of view, it must be taken into account that the user has to wait the system reply after giving feedback. This fact arises the  
65 need of any IPR system to assure a fast response process. If each feedback provided by the user results in a large amount of time before the system gives the next hypothesis, the process becomes very tedious and the classical, offline scenario may be preferred despite of needing more corrections. Unfortunately, seeking the most profitable hypothesis on our framework may take a high load  
70 of computation.

For all above, this work proposes an efficient approach to deal with these tasks. Once the problem is modeled within a weighted-graph-alike structure, the weights of this structure are efficiently recomputed so that each step of the ISPR scenario becomes linear with respect to the size of the uncorrected  
75 segment. Although this approach may consume additional resources at the very beginning of the task, it is worth when taking into account the whole interactive process. Experimentation shows that there are some heavy tasks that might become unfriendly if they are not approached with our proposal, which allows computing them more efficiently.

80 On the other hand, we assume that many of the SPR tasks have an underlying language model. The inclusion of a language model in the recognition process has demonstrated its utility in several works and applications [14, 15]. Within this context, Regular Languages (RL) are common ways to represent such models [16]. We will show in this paper that the proposed approach suits  
85 especially well when the language model is represented by an underlying RL since it allows modeling the problem with a better and more compact representation.

The rest of the paper is as follows: Section 2 provides background and problem statement of the ISPR framework stated here. Development of our  
90 approach is described in Section 3, which includes additional explanation of the RL case. Section 4 shows the experimentation, in which our approach is faced against the conventional search. Conclusions and future work are drawn in Section 5.

## 2. The ISPR framework

95 Let  $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_{|\Sigma|}\}$  be a non-empty set of symbols called *alphabet*. We call *sequence* each element  $w = w_1 w_2 \dots w_n$ ,  $w_i \in \Sigma$ . The (infinite) set of sequences is denoted as  $\Sigma^*$ . Let us use  $a, b, \dots$  for symbols and  $s, t, \dots$  for sequences. We denote the concatenation of  $s$  and  $a$  ( $s$  and  $t$ ) as  $sa$  ( $st$ ). Let  $s_{i\dots j}$  denote the sub-sequence of  $s$  from  $i$ th symbol to  $j$ th symbol (both inclusive).

100 We use  $\lambda$  to denote the empty sequence, which fulfills  $\lambda a = a = a\lambda$ .

The task of an SPR system is to guess the correct sequence  $\hat{s} \in \Sigma^*$  codified in each input received. Typically, the set of possible or allowed solutions ( $\Omega$ ) is a subset of the whole space ( $\Omega \subseteq \Sigma^*$ ).

The optimum strategy to reduce the error rate –assuming a *zero-one* loss function– is to propose the hypothesis  $\hat{h}$  for which the posterior probability  
 105 given input  $x$  is maximum (MaxPost) [17]:

$$\hat{h} = \arg \max_{h \in \Omega} \Pr(h|x) \quad (1)$$

We assume that, if a user is involved actively in the recognition process, the output is corrected sequentially until reaching the correct solution. Within the ISPR framework considered in this work, these corrections will consist in  
 110 providing the correct symbol of the first error found following a left-to-right order. Hence, at each iteration the system receives an error-free prefix  $t$  –that is,  $t$  is a prefix of the correct solution for the task– and proposes a new solution. The MaxPost strategy has to be modified to handle this new information:

$$\hat{h} = \arg \max_{h \in \Omega} \Pr(h|x, t) \quad (2)$$

This strategy is known to minimize the error rate after each user interaction.  
 115 Nevertheless, the main goal here is not to reduce error rate at each iteration but to reduce human effort thorough the whole process. If this human effort is measured as the number of user interactions needed to complete the task, the optimum criterion to minimize user corrections (MinCorr) can be computed by means of an incremental strategy. Let  $\Pr(u\Sigma^*) (= \sum_{s \in \Sigma^*} \Pr(us))$  denote the  
 120 probability of the prefix  $u$ . Then, the best hypothesis  $\hat{h} = \hat{h}_1 \cdots \hat{h}_n$  following MinCorr strategy is given by:

$$\hat{h}_i = \arg \max_{h_i \in \Sigma} \Pr(\hat{h}_{1 \dots i-1} h_i \Sigma^* | x, t) \quad (3)$$

There is no need of knowing the size of the output in advance, but the strategy stops when a particular symbol (*eg.* \$) is reached. This strategy was

developed by Oncina [12]. MinCorr constructs the output incrementally by  
 125 choosing at each step  $i$  just one label, which is concatenated to the accumulated  
 sequence  $\hat{h}_{1\dots i-1}$ . The label chosen is that which leads to the most probable  
 prefix. Reader is referred to the original work for further analysis and demon-  
 stration of optimality. Note that MaxPost strategy (Eq. 2) is actually different  
 to this optimum search, despite being commonly used as an approximation [18].

### 130 2.1. Modeling an ISPR task

This section describes a structure to model an ISPR task. We also present  
 the computation of the optimum strategy within this structure. The model  
 presented in this section will be used along the following sections.

Let us use an example to guide the explanation. Let us consider an SPR  
 135 task such that the vocabulary is  $\Sigma = \{a, b\}$  and the set of allowed solutions is  
 defined as  $\Omega = \{s : s \in \Sigma^* \text{ and } s \text{ does not contain two consecutive } b\}$ . At some  
 instance, an input  $x$  is received. For the sake of simplicity in the explanation,  
 we assume that input is perfectly segmented into three pieces  $x_1$ ,  $x_2$  and  $x_3$ .  
 Then, Table 2 shows a possible mapping of this input onto probabilities.

$\Sigma$	$\Pr(\cdot x_1)$	$\Pr(\cdot x_2)$	$\Pr(\cdot x_3)$
a	0.4	0.3	0.6
b	0.6	0.7	0.4

Table 2: Probability distribution over some input  $x = (x_1, x_2, x_3)$ .

140 For any given input, we assume that the solution space is restricted to a dis-  
 crete set of sequences ( $\{aaa, aab, aba, baa, bab\}$  in our running example). There-  
 fore, the task can be modeled with a weighted directed acyclic graph. In this  
 context, these graphs are usually referred to as Word Graphs [5] or Word Lat-  
 tices [19]. It is common to find this kind of structures in most SPR tasks such as  
 145 Machine Translation [20], Automatic Speech Recognition [21] or Handwritten  
 Text Recognition [22], since the solution space can be represented in a compact  
 way.

Given their widespread use, each author may customize these structures to fit the problem at hand. For example, the weights considered within these graphs are usually unconstrained. In this work, however, we assume that the weights represent probabilities and, therefore, the values must fall in the range  $[0, 1]$ .

**Definition 1.** A Word Graph (WG) is defined as a 6-tuple  $(V, A, \Sigma, p, v_0, F)$  such that:

- $V = \{v_0, \dots, v_{|V|}\}$  is the set of vertices.
- $\Sigma$  is the alphabet.
- $A \subseteq V \times \Sigma \times V$  is the set of edges. A restrictive condition holds over  $A$ , for which cycles are not allowed (neither loops nor cycles involving several edges).
- $P : A \rightarrow [0, 1]$  is the probability function over the edges.  $P((v, a, v')) = 0$  can be interpreted as “no transition from  $v$  to  $v'$  labeled with  $a$ ”.
- $v_0$  represents the initial vertex.
- $F \subseteq V$  is the set of accepting vertices. Without loss of generality, we assume that there are no out-transitions from accepting vertices.

A path is a sequence  $\pi = v_0 a_1 v_{i_1} a_2 \dots a_n v_{i_n}$ , where  $\sigma(\pi) = a_1 \dots a_n$ ,  $a_i \in \Sigma$ . The probability of a path is defined as follows:

$$P(\pi) = \prod_{i=0}^n P((v_{i_0}, a_{i+1}, v_{i_1})) \quad (4)$$

Let  $\Pi(s) = \{\pi : \sigma(\pi) = s\}$  denote the set of paths that output a symbol sequence  $s \in \Sigma^*$ . The probability of a symbol sequence is therefore computed by summing over the probabilities of all the paths of this set:

$$P(s) = \sum_{\pi \in \Pi(s)} P(\pi) \quad (5)$$



165 For instance, Fig. 2 shows a graphical representation of a WG that could  
have been built from our example input  $x$ . This is indeed a very simple ex-  
ample, which has been chosen in order to clarify the subsequent developments.  
Note, however, that if no assumption is considered about the vocabulary, the  
search space can become exponentially large because it is necessary to keep the  
170 history of each path in order to know whether the sequence must be accepted.  
Nevertheless, we will see below that the complexity of the WG can be reduced  
hugely if the vocabulary is defined by a Regular Language. This knowledge  
allows building a more compact WG by merging those vertices that represent  
equivalent histories and pruning those edges that lead to strings rejected by the  
175 language.

We say that  $\pi = v_0 a_1 v_{i_1} a_2 \dots a_n v_{i_n}$  is an *accepted path* if, and only if,  
 $v_{i_0} = v_0$  and  $v_{i_n} \in F$ . We will use  $\Pi_A$  to denote the set of accepted paths. Due  
to this task constraints, the probability of a path is computed as a function of  
the language model. For the sake of clarity, we will nullify a path depending  
180 whether or not it belongs to  $\Pi_A$ . We use  $\check{P}$  to denote this function, which is  
defined as:

$$\check{P}(\pi) = \begin{cases} \prod_{i=0}^n P((v_{i_0}, a_{i+1}, v_{i_1})), & \text{if } \pi \in \Pi_A \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

The case in which a path does not belong to the set of accepted sequences  
can be modeled by giving it a null *a priori* probability. Quite often, however,  
*a priori* probabilities are based on smoothed language models which assign a  
185 non-null probability to any sequence [23]. Either ways, what it is important to  
keep in mind is that the WG is not defining a proper probability distribution  
over the set of sequences. We shall revisit this issue later.

The problem after building such structure is how to choose a hypothesis to  
propose as solution to input  $x$ . For instance, MaxPost criterion would give the  
190 sequence  $bab$  because it is the one with the highest score. On the other hand,  
MinCorr would give  $aab$  (Table 3 shows the trace of this computation for the

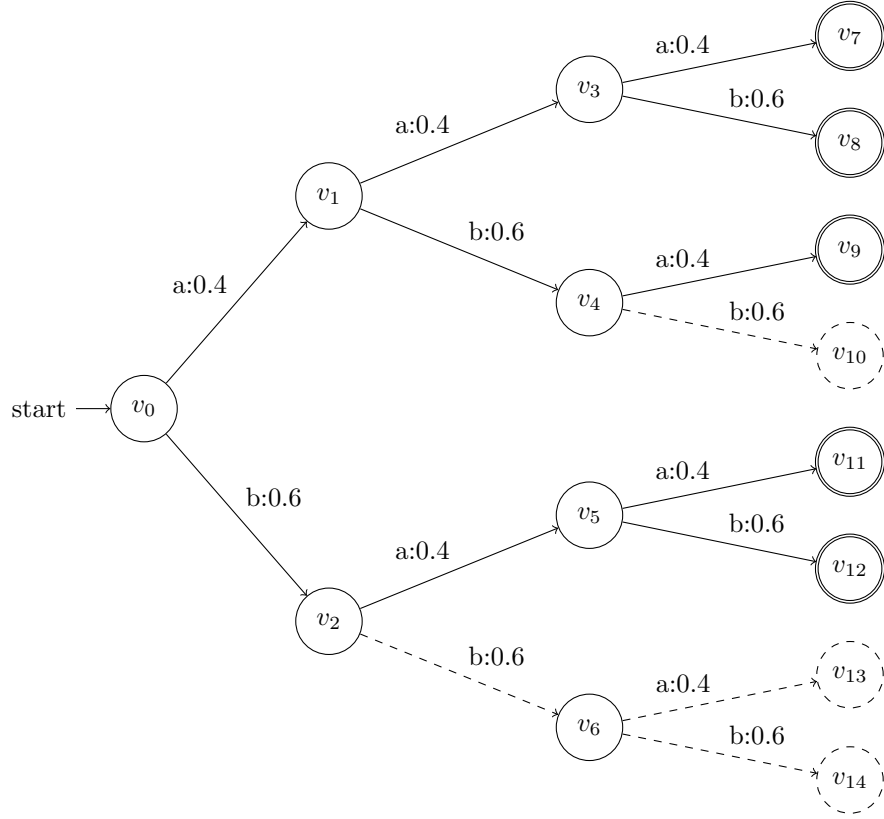


Figure 2: Graphical representation of the WG with respect to distribution of Table 2 and the set of allowed sequences.

first proposal).

Let us suppose that our input is represented by the WG showed above but the actual output sequence is *aaa*. Table 4 shows a complete trace of the interaction  
 195 between machine and user (which is devoted to giving the next error-free prefix), following both MinCorr and MaxPost criteria, to complete task assuming such output sequence.

In this case, MinCorr just needs one correction, whereas MaxPost needs three. However, what it is interesting to know is the expected number of cor-  
 200 rections required for each sequence (the number of corrections weighted by its probability), and taking into account the entire task, knowing what criterion has

i	$h_{1\dots i-1}$	$\check{P}(h_{1\dots i-1}a\Sigma^*)$	$\check{P}(h_{1\dots i-1}b\Sigma^*)$
1	$\lambda$	<b>0.256</b>	0.24
2	a	<b>0.16</b>	0.096
3	aa	0.064	<b>0.096</b>
4	aab	0	0

Table 3: Trace of the execution of the MinCorr criterion for the first proposal (no error-free prefix is known).

the minimum expected number of corrections. Table 5 depicts both the number of corrections and its expectancy obtained by each criterion assuming each of the allowed sequences. Statistics about the error rate are also included. Note  
205 that the probability of the sequence is given by the WG after normalizing them all to sum up to 1. As commented above, MinCorr is the optimum criterion when pursuing a minimum number of user corrections although MaxCorr is still the best option to reduce the error rate.

If we assume that our task is always modeled within a WG like that described here, computing MinCorr is equal to compute, from a current vertex  
210 at depth  $i$ , which of its out-transitions is able to reach more probability, *i.e.*  $\arg \max_{a \in \Sigma} \check{P}(h_{1\dots i-1}a\Sigma^*)$ .

In a trivial case, in which every single sequence belongs to the solution space ( $\Sigma^* = \Omega$ ), the probability of the edges would be exactly what it is needed. In  
215 this case, the probability reached following any vertex is 1, and the decision lies in the probability of the edges themselves. Thus the strategy becomes of linear complexity. Nevertheless, when some sequences are nullified (or its probability is weighted), it is unknown how much probability is achieved following an edge unless adding up the probabilities of all accepted paths that pass through that  
220 vertex. The underlying difference with respect to the trivial case is that  $\check{P}$  is not defining a proper probability because the sum over all the accepted sequences is not 1.

Under this circumstance, classical brute-force MinCorr computation requires

y = aaa			
MinCorr		MaxPost	
Machine (h)	aab	Machine (h)	bab
User prefix (p)	aaa	User prefix (p)	<b>a</b>
Machine (h)	aaa	Machine (h)	aba
User prefix (p)	#	User prefix (p)	<b>aa</b>
		Machine (h)	aab
		User prefix (p)	<b>aaa</b>
		Machine (h)	aaa
		User prefix (p)	#

Table 4: Trace of the human-computer interaction to complete the task if the actual sequence is *aaa*, considering both MinCorr and MaxPost criteria. # represents that user accepts the sequence.

an asymptotically exponential time with respect to the input. In the interactive framework proposed by Alabau et al. [24], computing the most probable prefix is reduced to a forward-backward computation over the graph. However, proposing a complete hypothesis needs to perform this computation several times. Worse still, after each user interaction this computation has to be repeated. Eventually, the whole process may become very inefficient.

Our proposal to solve this situation is to make the appropriate changes in the probabilities of the edges so that  $\tilde{P}$  becomes a canonical distribution. It is easy to see that the probability over paths after removing those that do not fulfill the constraints must be obtained by uniformly distributing the probability loss among all the accepted ones. In other words, the probability of an accepted path has to be multiplied by the inverse of the probability of all accepted paths.

		Corrections				Error			
		MinCorr		MaxPost		MinCorr		MaxPost	
y	p(y)	No.	Expect.	No.	Expect.	No.	Expect.	No.	Expect.
aaa	0.13	1	0.13	3	0.39	1	0.13	1	0.13
aab	0.19	0	0	2	0.38	0	0	1	0.19
aba	0.19	1	0.19	1	0.19	1	0.19	1	0.19
baa	0.20	2	0.40	1	0.20	1	0.20	1	0.20
bab	0.29	1	0.29	0	0	1	0.29	0	0
$\sum_y$	1		1.01		1.16		0.81		0.71

Table 5: Number of corrections (No.) and its expectancy (Expect.) for each of the allowed sequences of our running example considering both MinCorr and MaxPost criteria. Statistics about the error rate are also included.

That is,

$$P_N(\pi) = \frac{\check{P}(\pi)}{\sum_{\pi' \in \Pi_A} \check{P}(\pi')}, \forall \pi \in \Pi_A \quad (7)$$

Thus normalized probability distribution  $P_N$  over accepted paths would be obtained. Note that this is not relevant when dealing with a conventional SPR task: since all sequences are weighted by the same value, MaxPost criterion is not affected at all. This is why previous works focused on the non-interactive case might have not pay attention to the fact of not defining canonical probability distributions.

Finally, it is also important to stress that obtaining these new probabilities by a brute-force strategy entails exponential computation with respect to the depth of the WG. We propose in this work an approach to deal efficiently with this interactive scenario. Our idea is to build an alternative structure in which probabilities are normalized properly and efficiently. Although this may entail some additional cost at the very beginning of the task, we will show that using

this new structure constitutes a significant save during the whole ISPR process.

### 250 3. An efficient strategy for ISPR

In next lines we present an efficient approach to model an ISPR task so that the computation of the optimum criterion (MinCorr) becomes trivial. We propose to build a new WG, from now on referred as *normalized* WG, in which the probability distribution is readjusted as commented in Eq. 7.

255 This new structure can be built such that its topology is equal to the original WG that describes  $\check{P}$ , and the probabilities of the edges are obtained by making appropriate changes in the probabilities of the original edges. This process can be efficiently performed with the strategy described below.

Let  $(v, a, w) \in A$  be an edge of the WG that goes from vertex  $v$  to vertex  $w$  with label  $a$ , and let  $p((v, a, w))$  be the probability assigned to that edge. We 260 will denote as  $S(v)$  the achievable probability of a vertex, that is, the sum of the probabilities of all the paths that start at vertex  $v$ . A recursive definition of this function becomes:

$$S(v) = \begin{cases} 1, & \text{if } v \in F \\ \sum_{(v,a,w) \in A} P((v,a,w)) \cdot S(w), & \text{otherwise} \end{cases} \quad (8)$$

Note that, as (by definition) final vertices do not contain ongoing edges, the 265 probability that can be reached from a final vertex is 1, *i.e.*, no probability loss. In the rest of vertices, this probability is the sum of the probabilities achieved by following each possible path.

Now, the probability of each edge has to be recomputed. We fix the new probability of an edge, denoted as  $P_N$ , as its original one weighted by the achievable 270 probability of its destiny vertex. To satisfy probability constraints, the sum of the probabilities of the edges of a vertex must be 1 so all of them are divided by a normalization factor:

$$P_N((v, a, w)) = \frac{P((v, a, w)) S(w)}{\sum_{(v,a,w') \in A} P((v,a,w')) S(w')} = \frac{P((v, a, w)) S(w)}{S(v)} \quad (9)$$

This simple procedure is redistributing the probability loss caused by those sequences that are not accepted due to the problem constraints. What it is made is distribute this loss among all the accepted paths by assigning them a probability equal to its original one divided by the total probability. As mentioned above, when there is no loss, the total probability is 1 and this process would not change the probability of the edges.

It is trivial to check that those path that do not belong to  $\Pi_A$  have a null probability in  $P_N$ . Therefore, Eq. 4 and 6 are equal with respect to  $P_N$ . Now we have to show that the way  $P_N$  is calculated implies a well-defined distribution. That is,  $P_N(s) \geq 0 \forall s$  (evident) and  $\sum_s P_N(s) = 1$ .

**Lemma 1.** *The sum over the probabilities of all accepted paths in the normalized WG is 1:*

$$\sum_{\pi \in \Pi_A} P_N(\pi) = 1 \quad (10)$$

*Proof.* Let  $\pi = v_{\pi_0} a_1 v_{\pi_1} a_2 \dots a_n v_{\pi_n}$  denote an accepted path over the WG. Then,

$$\begin{aligned} \sum_{\pi \in \Pi_A} P_N(\pi) &= \sum_{\pi \in \Pi_A} \prod_{i=0}^{n-1} P_N((v_{\pi_i}, a_{i+1}, v_{\pi_{i+1}})) \\ &= \sum_{\pi \in \Pi_A} \check{P}(\pi) \frac{1}{S(v_{\pi_0})} \\ &= \sum_{\pi \in \Pi_A} \check{P}(\pi) \frac{1}{S(v_0)} \\ &= S(v_0) \frac{1}{S(v_0)} = 1 \end{aligned} \quad (11)$$

285

□

We also have to show that  $P_N$  models what it is pursued (Eq. 7).

**Lemma 2.** *The new probability distribution  $P_N$  of the WG holds that*

$$P_N(\pi) = \frac{\check{P}(\pi)}{\sum_{\pi' \in \Pi_A} \check{P}(\pi')}, \forall \pi \in \Pi_A \quad (12)$$

*Proof.* Let  $\pi = v_{\pi_0} a_1 v_{\pi_1} a_2 \dots a_n v_{\pi_n}$  be an accepted path of the WG. The probability of  $\pi$  is originally computed as

$$\check{P}(\pi) = \prod_{i=0}^{n-1} p((v_{\pi_i}, a_{i+1}, v_{\pi_{i+1}})) \quad (13)$$

The probability of the same path in  $P_N$  is

$$\begin{aligned} P_N(\pi) &= \prod_{i=0}^{n-1} P_N((v_{\pi_i}, a_{i+1}, v_{\pi_{i+1}})) \\ &= \prod_{i=0}^{n-1} P((v_{\pi_i}, a_{i+1}, v_{\pi_{i+1}})) \frac{S(v_{\pi_{i+1}})}{S(v_{\pi_i})} \\ &= \check{P}(\pi) \left( \prod_{i=0}^{n-1} \frac{S(v_{\pi_{i+1}})}{S(v_{\pi_i})} \right) \\ &= \check{P}(\pi) \frac{S(v_{\pi_n})}{S(v_{\pi_0})} \\ &= \check{P}(\pi) \frac{1}{S(v_{\pi_0})} \\ &= \check{P}(\pi) \frac{1}{\sum_{\pi' \in \Pi_A} \hat{p}(\pi')} \end{aligned} \quad (14)$$

Because of the WG definition, every accepted path starts in the same vertex ( $v_0$ ) so the value of  $S(v_{\pi_0})$  is the same for all of them. Therefore, it equals the sum of the probabilities of all accepted paths due to Eq. 8.  $\square$

290 The previous normalization process can be performed efficiently over the WG itself by following a Dynamic Programming scheme, as explained in [25]. Figure 3 shows the result of this process for the WG given in Fig. 2. One can check that probabilities of the edges match the normalized values showed in the trace of MinCorr criterion (see Table 3).

295 With this new probability distribution over the paths, the optimum hypothesis can be built by just choosing at each vertex the edge with the highest probability. Although this approach is greedy, after the normalization process the complexity is reduced to a linear computation as we have to go across the depth of the WG only once.



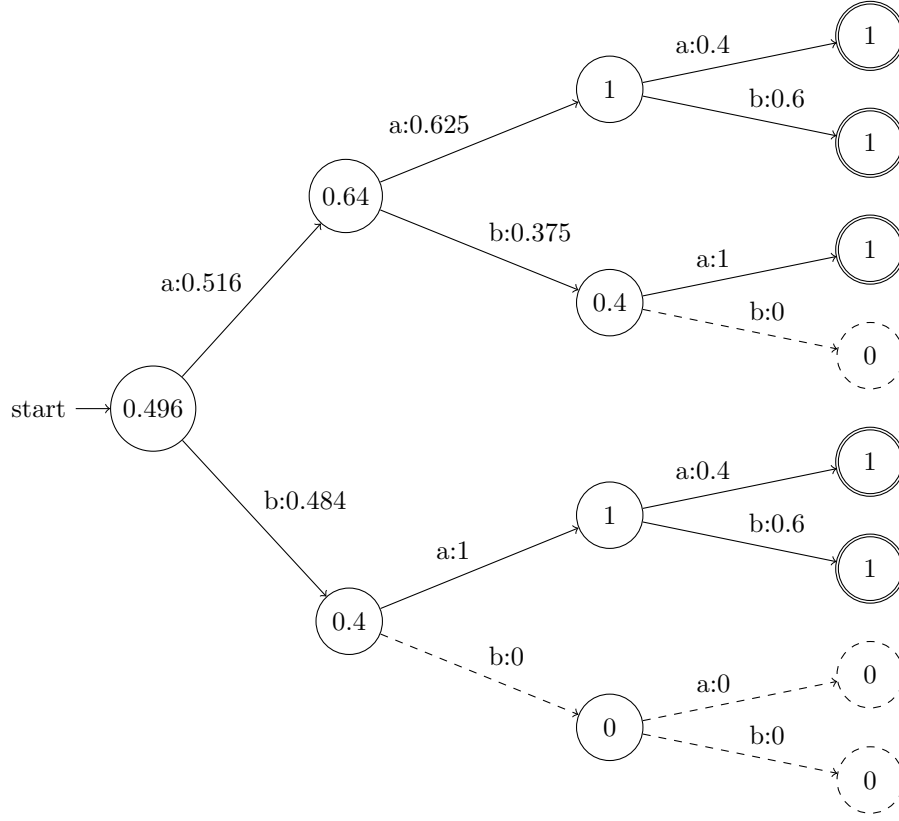


Figure 3: Example of WG after normalization. Values inside the vertices represent  $S$  function.

### 3.1. Special case: ISPR with RL

Next lines describe how the efficient approach presented in the previous section suits especially well when we assume that any accepted sequence belongs to a certain regular language  $R$ —that is,  $\Omega \equiv L(R)$ —. If this RL is known, we can take advantage of any of its equivalent models such as regular expressions or finite-state automata to improve the performance of the system. Our approach makes use of a Deterministic Finite-State Automaton representation of such languages.

**Definition 2.** A Deterministic Finite-State Automaton (DFA) can be defined as a 5-tuple  $(Q, \Sigma, \delta, q_0, F)$  where

- $Q$  is the set of states

- $\Sigma$  is the alphabet
- $\delta : Q \times \Sigma \rightarrow Q$  is the transition function
- $q_0$  is the initial state
- $F \subseteq Q$  is the set of accepting states

315 A sequence  $s$  belongs to the language defined by the DFA if, and only if,  
 $\delta^*(q_0, s) \in F$ .

The task can again be modeled on a WG like that defined in the previous section. However, taking advantage of the DFA, the construction of the structure can be reduced to Algorithm 1. This operation is usually referred as  
 320 *composition* [26].

The algorithm is devoted to building a graph with vertices  $V = \{v_0, \dots, v_{|V|}\}$ . It generates as many vertices as possible combinations of states of the automaton ( $|Q|$ ) and positions in the input sequence ( $n+1$ ). Let  $v_{i,j}$  denote the vertex that represents the joint of the  $i$ th state of the automaton and the  $j$ th position  
 325 of the sequence. Each arc is denoted as a 3-tuple  $(s, \sigma, d)$  with  $s \in V$  being the source vertex,  $\sigma \in \Sigma$  being the emission symbol and  $d \in V$  being the destiny vertex. The probability of such arc is defined as  $P((s, \sigma, d)) \in [0, 1]$ . Basically, the algorithm joins each pair of vertices  $(v_{i,j}, v_{k,j+1})$  with an emission symbol  $\sigma$  if, and only if,  $\delta(i, \sigma) = k$ . The associated probability is given by the value  
 330  $\Pr(\sigma_m | x_j)$ . Finally, a vertex belongs to  $F$  if, and only if, it represents the joint of a final state of the original WG and an accepting state of the DFA.

The complexity of this procedure is  $O(|Q|n)$ . To better illustrate its behavior, let us show a simple example: given the DFA of Fig. 4 and the probability distribution showed in Table 6, a WG like that of Fig. 5 is obtained (dashed  
 335 elements can be removed after pruning unreachable and rejected states). The vertex  $q_{i,j}$  represents the state  $q_i$  of the automaton in the position  $j$  of the input sequence.

As mentioned at the beginning of this section, our approach suits perfectly in this case. Note that this WG contains the minimum number of vertices needed

**Data:**  $(Q, \Sigma, \delta, q_0, F_D) : \text{DFA}, \Pr(\cdot|x), n : \mathbb{N}$

**Result:**  $(V, A, p, v_0, F_P) : \text{WG}$

$V = \{1, \dots, |Q|\} \times \{1, \dots, n\};$

$v_0 = V_{0,0};$

$A = \emptyset;$

**for**  $i \in Q$  **do**

**for**  $j = 1 : n + 1$  **do**

**for**  $\sigma \in \Sigma$  **do**

$A = A \cup (v_{i,j}, \sigma, v_{\delta(i,\sigma),j+1});$

$P((v_{i,j}, \sigma, v_{\delta(i,\sigma),j+1})) = \Pr(\sigma|x_j);$

**end**

**end**

**end**

$F_P = \{v_{i,j} : j = n + 1, q_i \in F_D\}$

**Algorithm 1:** Building a WG modeling the problem from the DFA.

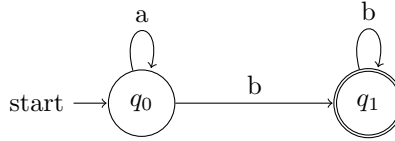


Figure 4: Example of DFA defining the RL  $a^*bb^*$

$\Sigma$	$\Pr(\cdot x_1)$	$\Pr(\cdot x_2)$	$\Pr(\cdot x_3)$	$\Pr(\cdot x_4)$
a	0.4	0.7	0.2	0.3
b	0.6	0.3	0.8	0.7

Table 6: Example of probability distribution over some input  $x = (x_1, x_2, x_3, x_4)$ .

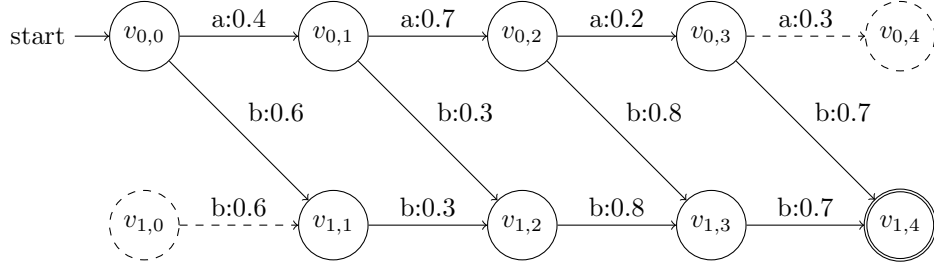


Figure 5: Example of WG from DFA of Fig. 4 and probability over the input of Table 6.

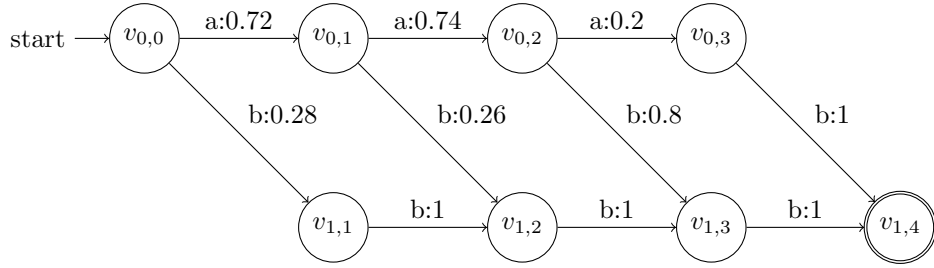


Figure 6: WG of Fig. 5 after normalization.

340 to model the problem as long as the DFA is the minimum one to represent the language.

Once the problem is modeled on the WG, it can be normalized efficiently as described in the previous section. For instance, Fig. 6 shows the result of applying this normalization over the WG of Fig. 5.

345 It must be kept in mind that this approach can also be applied when the output is defined by a Deterministic Push-Down Automata (DPDA), which defines a subset of the Context-Free Languages (Deterministic Context-Free Languages). The difference is that the possible vertices in this WG would be defined by both the number of states of the DPDA and the size of the input as well as the possible configurations of the stack, since each combination of DPDA state and content  
350 of the stack has to be considered as a single state.

## 4. Experimentation

The experimentation section is aimed mainly at showing the complexity reduction achieved by using our approach. This complexity will be measured as the time needed by the machine to give the first hypothesis <sup>1</sup>. This indicator gives a maximum time for single proposals since the first hypothesis is expected to be which more computation requires. When using our proposal, this time will include the normalization process.

Thorough all the experiments, the input is a sequence of features and the task consists in guessing the correct output sequence. If the sequence proposed by the machine is incorrect, the shortest correct prefix of the solution is given to the system before recomputing the next hypothesis. This way we simulate a user correcting the first error found, as assumed in our scenario. Note that user interactions are simulated in order to perform a comprehensive experimentation, as done in previous works [18].

We will compare our approach against that based on a dynamic programming Forward-Backward (DP-FB) computation [24]. In that case, the probability of a prefix  $sa$  is given by

$$P(sa\Sigma^*) = \sum_{(v,w) \in V^2} \phi_s(v) P((v,a,w)) \beta(w) \quad (15)$$

where  $\phi_s(v)$  represents the probability of parsing from the initial vertex to vertex  $v$  (forward computation) taking into account only those paths that match  $s$ . On the other hand,  $\beta(v)$  denotes the probability that can be achieved from vertex  $v$  to the final vertices (backward computation). On the contrary, the brute-force strategy has been avoided in this comparison since it is clearly too inefficient.

### 4.1. Experimentation with synthetic data

In the first experiment, synthetic data will be used. For this case, we assume that our solution space is given as a list of sequences and not in the form of

---

<sup>1</sup>The machine used in all the experiments was an Intel Core i5-2400 CPU at 3.10GHz with 4 GB of RAM

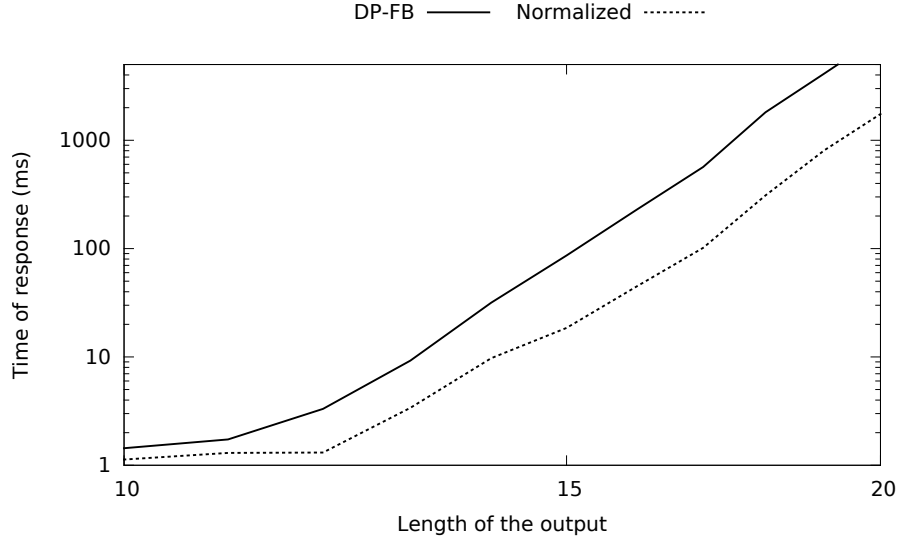


Figure 7: Time elapsed (milliseconds) for computing the first hypothesis with our normalization (Normalized) of the WG and with the dynamic programming Forward-Backward (DP-FB) scheme. Values presented are obtained averaging 1000 executions.

regular language. Although each finite set of sequences can be modeled as a  
 375 regular language (by their union), it is not interesting for our experimentation  
 to see it in that way.

Let us use an alphabet that consist of just two symbols and let  $\Omega$  consist of  
 a random subset of the set of possible sequences ( $2^n$ ). In this experiment, we es-  
 tablished that at least 75% of the complete set is included. For the experiments,  
 380 we first generate randomly the probability distribution of each symbol in each  
 place of the sequence. Then, the sequence that must be guessed is randomly  
 chosen accordingly to that distribution.

Figure 7 shows the average response time for computing the first hypothesis  
 with our normalization of the WG and with the DP-FB computation with respect  
 385 to the size of the input. Although curves reflect the exponential growth of the  
 WG, results report a noticeable difference of time between these approaches as  
 the size of the task is increased. The curves of the normalization approach show  
 a noticeable lower slope, reflecting the reduction of time complexity pursued.

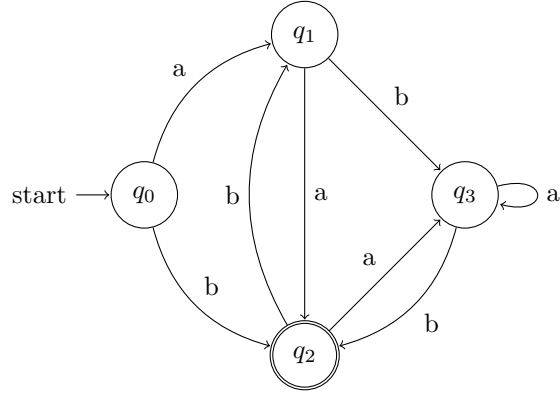


Figure 8: A toy DFA for the synthetic experimentation

#### 4.2. Experimentation with synthetic data: RL case

390 Let us establish a setup like that of the previous experiment. Nevertheless, let us now suppose that an underlying RL is defining our solution space. Any form of this knowledge can be put into a DFA. For instance, let us use the (rather small) synthetic automaton of Fig. 8.

This DFA can be used to define the solution space  $\Omega$  as the sequences of such size that belongs to the language. However, we can model both the input  
 395 and the DFA in the same WG following the procedures explained in Section 3.1. These two ways of approaching the task (with or without RL) will be confronted experimentally. In addition, both of them with and without normalizing the resulting WG will be analyzed as well.

400 Average time results of the four combinations considered after running 1000 repetitions with different probability distributions over the strings are shown in Fig. 9. Note that both  $x$ -axis and  $y$ -axis are represented in a logarithmic scale.

As introduced before, taking into account the RL is the most relevant factor for an efficient computation. Results show that large sequences can only be  
 405 handled by building the WG in combination with the RL. Note, however, that even in that case the normalization procedure is able to improve the efficiency with respect to the dynamic programming approach, yet to a much lesser order of magnitude.

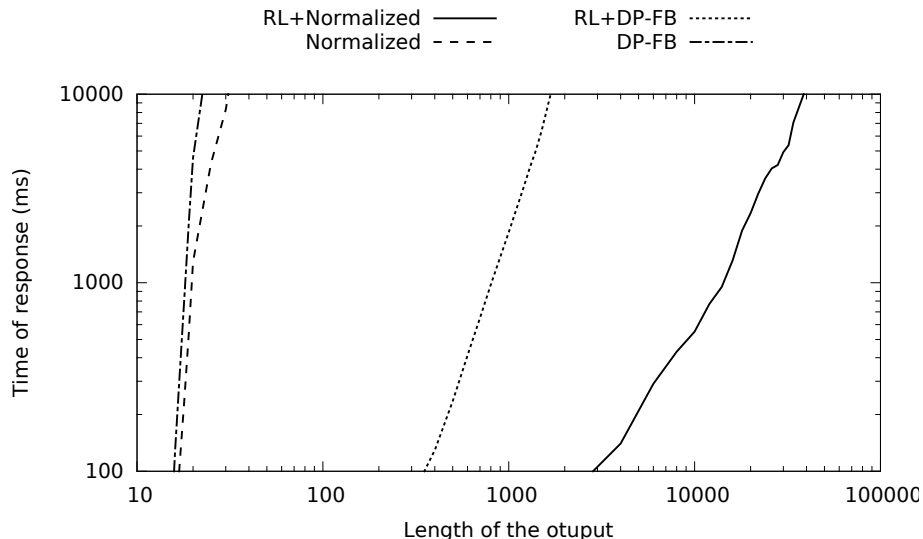


Figure 9: Time elapsed (milliseconds) for computing the first hypothesis with our normalization (Normalized) of the WG and with the dynamic programming Forward-Backward (DP-FB) scheme considering or not the construction with a Regular Language (RL). Values presented are obtained averaging 1000 executions.

#### 4.3. Online Optical Music Recognition case

410 The last experiment is carried out on a real task of Optical Music Recognition (OMR). The task of OMR is quite similar to the Optical Character Recognition with music symbols instead of alphanumeric characters [27, 28]. From the point of view of our experimentation, this task is highly interesting because musical sequences can be of an arbitrary length, as the number of bars is not limited.

415 In addition, given a time signature, the set of allowed musical symbols within a bar can be defined by a DFA. We will restrict ourselves to the use of a DFA modeling a *Common Time* time signature ( $\frac{4}{4}$  or **C**) so that the size of the WG depends only on the size of the input sequence. Specifically, this DFA consists of 220 states and 861 transitions.

420 As in the OCR field, the task can be applied to either data extracted from images (offline modality) or pen-based collected strokes (online modality). We will take advantage of the Handwritten Online Musical Symbols (HOMUS) database





Figure 10: An example of rendered input consisting of the sequence (*G Clef, 4 4 Time, Half Note, Quarter Note, Barline, Whole Note, Barline*).

[29] to set an online OMR task. The HOMUS consists of 15200 samples from different writers, each one providing 152 symbols spread over 32 common musical  
 425 symbol classes.

In addition to the computational time, we will confront experimentally the comparison between interactive and classical (non-interactive) scenario in terms of user corrections. We intend to emphasize the human effort reduction that can be achieved by using the interactive framework and how this reduction  
 430 may require the approach proposed in this paper. The interactive scenario will be addressed with the MinCorr criterion. In the non-interactive scenario, the problem will be tackled using the MaxPost criterion since it is the optimum way to reduce the error rate. Nevertheless, both scenarios will be evaluated by counting the number of corrections needed in each of them. Note that in the  
 435 non-interactive case, the number of corrections needed is just the number of errors in the first hypothesis proposed, which would be the case in a classical post-editing scenario.

At each execution, a random sequence is extracted by a random walk over the DFA. Each symbol of the sequence is substituted by test samples from the  
 440 HOMUS and the rest of the samples are used as training set. Figure 10 shows an example of image, rendered from the input stroke sequence, whose actual output sequence is (*G Clef, 4 4 Time, Half Note, Quarter Note, Barline, Whole Note, Barline*).

#### 4.3.1. Segmentation-driven scenario

445 We first consider an scenario in which it is known how to group strokes to form single musical symbols. Thus, the task can be solved by assigning one label to each group of strokes, as long as the language model is fulfilled.

To map each series of strokes onto symbol probabilities, the Nearest Neighbor rule will be used as non-parametric estimator [30]. The dissimilarity used is 450 Dynamic Time Warping (DTW), given its good results for classifying these samples [29]. To map an input  $x$  into probabilities of being each of the possible symbols of the task ( $\Sigma$ ), we resort to the following equation:

$$p(w|x) = \frac{1}{\min_{x' \in T_w} d(x, x')^n + \epsilon} \frac{1}{\sum_{w \in \Sigma} \frac{1}{\min_{x' \in T_w} d(x, x')^n + \epsilon}}, \forall w \in \Sigma \quad (16)$$

where  $T_w$  is the training set for symbol  $w$  and  $\epsilon$  is a non-zero value provided to avoid infinity values. Value  $n$  is a parameter that defines the *peakness* of the 455 probability. In our case, it was fixed to 10 based on a preliminary experimentation. The second term is just a normalization factor to make probabilities sum up to 1.

Figure 11 reports the recognition results of these experiments. It can be seen the profit of using an interactive scenario against the classical one in which 460 the task is completed offline. These results clearly support the use of the IPR scenario whenever it is possible.

Concurrently, DP-FB and normalized approaches to compute the MinCorr criterion have been compared in terms of time response. In this case, only those which include the RL are used since the previous experiment showed that they 465 are the most efficient ones. These computational time results are plotted in Fig. 12. One can see that the size of the WG affects noticeably to the complexity of the task. The bigger the graph, the higher the time of response. It is important to stress that the task become unpractical for the greatest automaton, even with sequences relatively small, when no normalization is performed. 470 In turn, the strategy that normalizes the WG obtain a faster time of response in all cases considered.

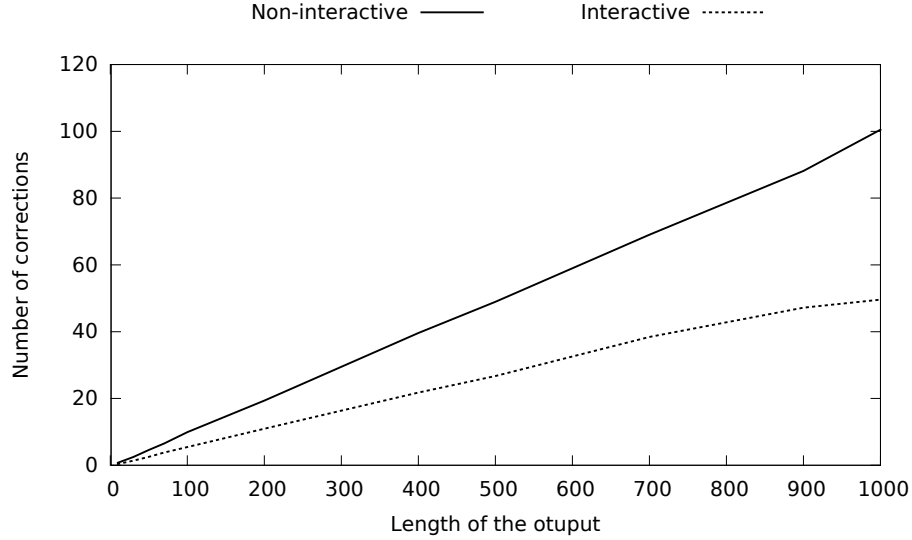


Figure 11: Segmentation-driven scenario: average number of user corrections needed to complete the recognition task comparing interactive and non-interactive post-editing scenarios. Values presented are obtained averaging 1000 executions.

These results reflect that a profitable way of facing this interactive task is by using the strategy developed in this paper, which allows a more efficient computation.

475 It should be emphasized that this task would be very inefficient even for short musical sequences without the RL-based approach proposed since the solution space would be of order  $32^n$  (being  $n$  the size of the sequence).

#### 4.4. Segmentation-free scenario

480 Another point of view of the same task, much closer to a real system, is that in which the input is not segmented into symbols but a sequence of strokes is received. Note that strokes are easily isolated because they are bounded by *pen-up* and *pen-down* actions of the e-pen.

In this case, the vertices of our WG would represent strokes (the  $i$ th vertex indicates the state after computing from the first input stroke to the  $i$ th one).  
 485 The point here is that the same musical symbol can be represented by several

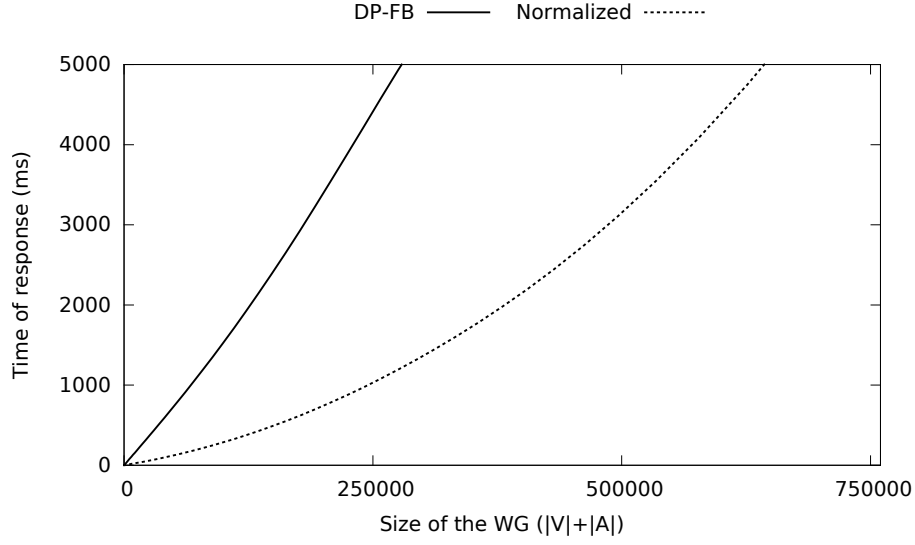


Figure 12: Segmentation-driven scenario: time elapsed (milliseconds) for computing the first hypothesis with our normalization (Normalized) of the WG and with the dynamic programming Forward-Backward (DP-FB) scheme. Values presented are obtained averaging 1000 executions.

strokes sequences which can even vary in number. For instance, a *Quarter Note* (♩) can be a *black note head* followed by a *stem* (Fig. 13a), or just the *quarter note* primitive if the symbol was written with a single stroke (Fig. 13b).



Figure 13: *Quarter Note* written with different sets of strokes.

This fact causes our WG to be no longer deterministic, as different edges with the same musical symbol can go from one vertex to many others as long as the underlying stroke sequence is different. Our strategy is changed so that computing a prefix is now done by taking into account all the paths that match such prefix. Furthermore, this scenario entails higher WGs since the number of

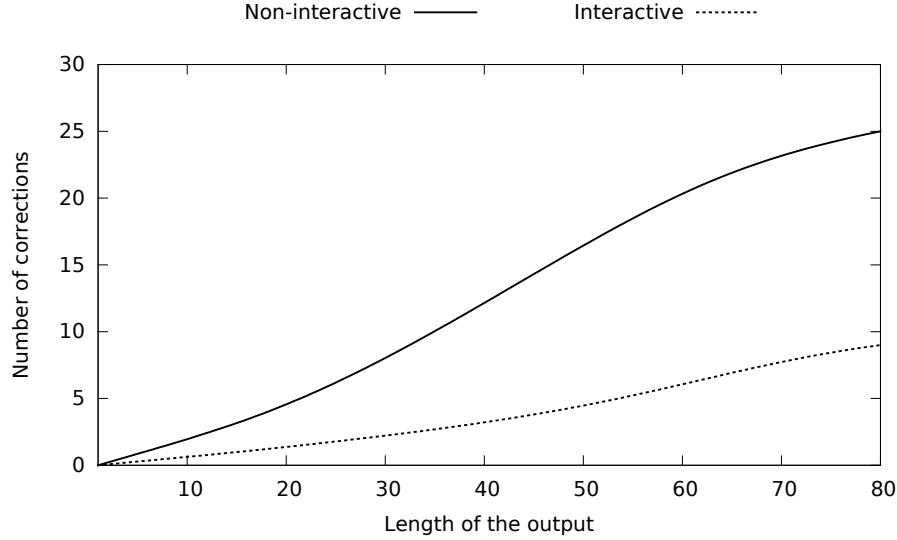


Figure 14: Segmentation-free scenario: average number of user corrections needed to complete the recognition task comparing interactive and non-interactive post-editing scenarios. Values presented are obtained averaging 1000 executions.

strokes is always higher than the number of musical symbols.

495 We make use of previous works already done on this task to learn the set of stroke primitives and the sequence of stroke primitives that define each musical symbol [31, 32]. In that case, the probabilities of the edges are computed from the probability of each input stroke to be each of the considered stroke primitives. As in the previous case, it is done following a Nearest Neighbor probability estimation with DTW, with the peakness of the probability (parameter  $n$ ) fixed to 15.

We repeat the comparison between the number of corrections that would be necessary in both an interactive scenario and in a conventional, non-interactive post-editing one. Results are illustrated in Fig. 14. We again see that using an interactive approach involves a noticeable decrease in the number of corrections to be made, thereby reducing the expected human effort thorough the task.

Moreover, Fig. 15 shows the time that takes the computation of the first hypothesis comparing DP-FB against the proposed approach. It can be seen that

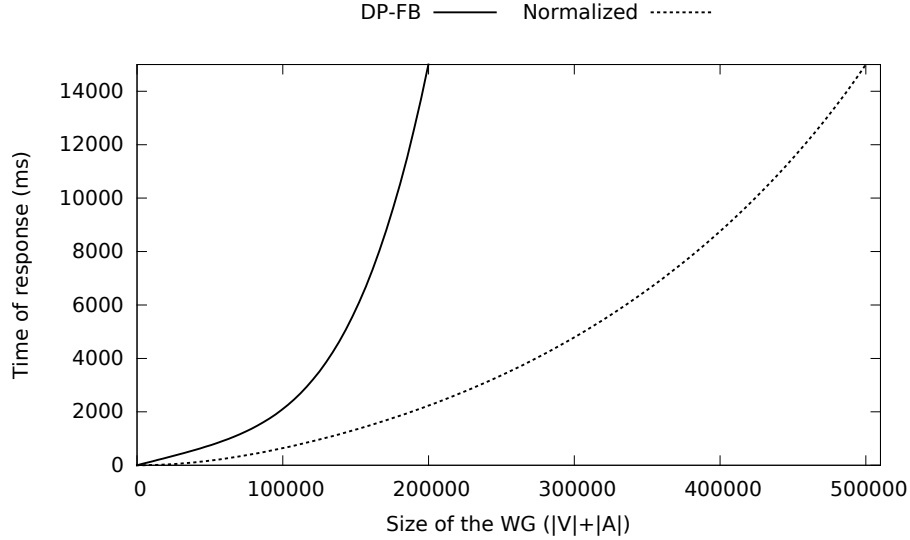


Figure 15: Segmentation-free scenario: time elapsed (milliseconds) for computing the first hypothesis with our normalization (Normalized) of the WG and with the dynamic programming Forward-Backward (DP-FB) scheme. Values presented are obtained averaging 1000 executions.

the proposed approach is still a more efficient strategy, although less pronounced  
510 in this case.

## 5. Conclusions

We have presented in this work an efficient approach to deal with IPR tasks. We focused on IPR tasks with sequential nature (the solution is a sequence of symbols). Over this framework the main problem is that the optimum criterion,  
515 which is expected to reduce the user corrections needed, may require a high computation. Since these tasks need an user to work with the machine, the process must be fast enough to not become user-unfriendly.

We considered modeling these problems with a WG and perform a normalization procedure. In addition, if the task is defined by a known Regular Language,  
520 we proposed a construction that takes into account this information to build a more compact WG.

Several experiments with synthetic and real data have been performed. Results have reported improvements in terms of time, making more manageable some heavy tasks. With respect to recognition accuracy, results of the interactive framework have shown a noticeable reduction of human effort (corrections) needed. In our experiments, some of these improvements are more profitable if our strategy is used since they are slower otherwise. Therefore, our strategy has proved to be very valuable for the ISPR framework. Results have reported that taking into account a RL results in the largest reduction of complexity (both in time and space), since the solution space can be modeled in a more compact structure. This fact has shown a higher relevance than the normalization procedure itself.

As future work it is intended to find a similar approach to deal with tasks defined by Context-Free Languages (CFL). This class of languages is known to define several interesting tasks. Note that the scenario set in this work consists of guessing an output of a discrete length. Thus, although knowing that each output has to belong to a CFL, for each instance of the task our solution space is indeed defined by a RL since each finite language belongs to such class of languages. Therefore, the problem could be modeled in some way on a WG as done in this work.

## Acknowledgments

Authors would like to thank the anonymous reviewers for their constructive comments to improve the quality of this work.

This work was partially supported by the Spanish Ministerio de Educación, Cultura y Deporte through FPU fellowship (AP2012-0939) and the Spanish Ministerio de Economía y Competitividad through Project TIMuL (No. TIN2013-48152-C2-1-R, supported by UE FEDER funds).

## References

- [1] A. Graves, A.-R. Mohamed, G. Hinton, Speech recognition with deep  
550 recurrent neural networks, in: Acoustics, Speech and Signal Processing  
(ICASSP), 2013 IEEE International Conference on, 2013, pp. 6645–6649.
- [2] F. Yin, Q.-F. Wang, X.-Y. Zhang, C.-L. Liu, ICDAR 2013 Chinese Hand-  
writing Recognition Competition, in: Document Analysis and Recognition  
(ICDAR), 2013 12th International Conference on, 2013, pp. 1464–1470.
- 555 [3] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, A. Klapuri, Automatic  
music transcription: challenges and future directions, *Journal of Intelligent  
Information Systems* 41 (3) (2013) 407–434.
- [4] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling,  
C. Monz, P. Pecina, M. Post, H. Saint-Amand, et al., Findings of the 2014  
560 workshop on statistical machine translation, in: Proceedings of the Ninth  
Workshop on Statistical Machine Translation, Association for Computa-  
tional Linguistics Baltimore, MD, USA, 2014, pp. 12–58.
- [5] A. H. Toselli, E. Vidal, F. Casacuberta, *Multimodal Interactive Pattern  
Recognition and Applications*, Springer, 2011.
- 565 [6] B.-Å. Lundvall, *National systems of innovation: Toward a theory of inno-  
vation and interactive learning*, Vol. 2, Anthem Press, 2010.
- [7] E. Horvitz, Principles of mixed-initiative user interfaces, in: Proceedings of  
the SIGCHI conference on Human Factors in Computing Systems, ACM,  
1999, pp. 159–166.
- 570 [8] E. Vidal, L. Rodríguez, F. Casacuberta, I. García-Varea, Interactive  
Pattern Recognition, in: *Machine Learning for Multimodal Interaction  
(MLMI)*, 2007, pp. 60–71.
- [9] V. Romero, J. Andreu Sanchez, Human Evaluation of the Transcription  
Process of a Marriage License Book, in: 12th International Conference on  
575 Document Analysis and Recognition (ICDAR), 2013, pp. 1255–1259.



- [10] G. Sanchis-Trilles, V. Alabau, C. Buck, M. Carl, F. Casacuberta, M. García-Martínez, U. Germann, J. González-Rubio, R. L. Hill, P. Koehn, et al., Interactive translation prediction versus conventional post-editing in practice: a study with the CasMaCat workbench, *Machine Translation* 28 (3-4) (2014) 217–235.
- [11] V. Romero, A. Fornés, N. Serrano, J.-A. Sánchez, A. H. Toselli, V. Frinken, E. Vidal, J. Lladós, The ESPOSALLES database: An ancient marriage license corpus for off-line handwriting recognition., *Pattern Recognition* 46 (6) (2013) 1658–1669.
- [12] J. Oncina, Optimum algorithm to minimize human interactions in sequential Computer Assisted Pattern Recognition, *Pattern Recognition Letters* 30 (5) (2009) 558–563.
- [13] J. Oncina, E. Vidal, Interactive Structured Output Prediction: Application to Chromosome Classification, *Pattern Recognition and Image Analysis (LNCS)* 6669 (2011) 256–264.
- [14] E. Roche, Y. Shabes (Eds.), *Finite-State Language Processing*, MIT Press, Cambridge, MA, USA, 1997.
- [15] F. Zamora-Martínez, V. Frinken, S. España-Boquera, M. Castro-Bleda, A. Fischer, H. Bunke, Neural network language models for off-line handwriting recognition, *Pattern Recognition* 47 (4) (2014) 1642 – 1652.
- [16] M. Mohri, *Finite-state Transducers in Language and Speech Processing*, *Comput. Linguist.* 23 (2) (1997) 269–311.
- [17] R. O. Duda, P. E. Hart, D. G. Stork, *Pattern Classification*, 2nd Edition, John Wiley & Sons, New York, NY, 2001.
- [18] A. H. Toselli, V. Romero, M. Pastor, E. Vidal, Multimodal interactive transcription of text images, *Pattern Recognition* 43 (5) (2010) 1814–1825.

- [19] C. Dyer, S. Muresan, P. Resnik, Generalizing word lattice translation, ACL-08: HLT (2008) 1012.
- [20] N. Ueffing, F. J. Och, H. Ney, Generation of word graphs in statistical machine translation, in: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10, EMNLP '02, Association for Computational Linguistics, Stroudsburg, PA, USA, 2002, pp. 156–163.
- [21] R. Justo, A. Pérez, M. Torres, Impact of the approaches involved on word-graph derivation from the asr system, in: J. Vitrià, J. Sanches, M. Hernández (Eds.), Pattern Recognition and Image Analysis, Vol. 6669 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 668–675.
- [22] V. Romero, A. H. Toselli, E. Vidal, Multimodal Interactive Handwritten Text Transcription, Series in Machine Perception and Artificial Intelligence (MPAI), World Scientific Publishing, 2012.
- [23] R. Kneser, H. Ney, Improved backing-off for m-gram language modeling, in: 1995 International Conference on Acoustics, Speech, and Signal Processing, ICASSP '95, Detroit, Michigan, USA, May 08-12, 1995, 1995, pp. 181–184.
- [24] V. Alabau, A. Sanchis, F. Casacuberta, On the optimal decision rule for sequential interactive structured prediction, Pattern Recognition Letters 33 (16) (2012) 2226 – 2231.
- [25] M. Pastor, J. A. Sánchez, A. H. Toselli, E. Vidal, Handwritten Text Recognition: Word-Graphs, Keyword Spotting and Computer Assisted Transcription, Tutorials of the 14th International Conference on Frontiers in Handwriting Recognition (2014).
- [26] M. Mohri, F. Pereira, M. Riley, Weighted finite-state transducers in speech recognition, Computer Speech & Language 16 (1) (2002) 69–88.

- 630 [27] D. Bainbridge, T. Bell, The challenge of optical music recognition, *Computers and the Humanities* 35 (2) (2001) 95–121.
- [28] A. Rebelo, I. Fujinaga, F. Paszkiewicz, A. R. S. Marçal, C. Guedes, J. S. Cardoso, Optical music recognition: state-of-the-art and open issues, *IJMIR* 1 (3) (2012) 173–190.
- 635 [29] J. Calvo-Zaragoza, J. Oncina, "Recognition of Pen-Based Music Notation: the HOMUS dataset, in: *Proceedings of the 22nd International Conference on Pattern Recognition*, Stockholm, Sweden, 2014, pp. 3038–3043.
- [30] K. Fukunaga, *Introduction to Statistical Pattern Recognition* (2Nd Ed.), Academic Press Professional, Inc., San Diego, CA, USA, 1990.
- 640 [31] J. Calvo-Zaragoza, J. Oncina, Clustering of strokes from pen-based music notation: An experimental study, in: R. Paredes, J. S. Cardoso, X. M. Pardo (Eds.), *7th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, Springer, Santiago de Compostela, Spain, 2015, pp. 633–640.
- 645 [32] J. Calvo-Zaragoza, J. Oncina, Recognition of pen-based music notation with finite-state machines, *Expert Systems with Applications* doi: 10.1016/j.eswa.2016.10.041.